Personal and Ubiquitous Computing
https://doi.org/10.1007/s00779-018-1122-9

ORIGINAL ARTICLE

A comprehensive reputation assessment framework for volunteered geographic information in crowdsensing applications

Nafaâ Jabeur[1] · Roula Karam[2] · Michele Melchiori[2] · Chiara Renso[3]

Received: 30 September 2017 / Accepted: 8 February 2018

Abstract
Volunteered geographic information (VGI) is the result of activities where individuals, supported by enabling technologies, behave like physical sensors by harvesting and organizing georeferenced content, usually in their surroundings. Both researchers and organizations have recognized the value of VGI content, however this content is typically heterogeneous in quality and spatial coverage. As a consequence, in order for applications to benefit from it, its quality and reliability need to be assessed in advance. This may not be easy since, typically, it is unknown how the process of collecting and organizing the VGI content has been conducted and by whom. In the literature, various proposals focus on an indirect process of quality assessment based on reputation scores. Following this perspective, the present paper provides as main contributions: (i) a multi-layer architecture for VGI which supports a process of reputation evaluation; (ii) a new comprehensive model for computing reputation scores for both VGI data and contributors, based on direct and indirect evaluations expressed by users, and including the concept of data aging; (iii) a variety of experiments evaluating the accuracy of the model. Finally, the relevance of adopting this framework is discussed via an applicative scenario for recommending tourist itineraries.

Keywords Social sensors · Reputation evaluation · Feedback-based model · Indirect feedback · Volunteered geographic information · Mobile crowdsourcing · Tourism planning

1 Introduction

The concept of citizens as sensors [9] refers to the idea that individuals, supported by emergent pervasive technologies, including smartphones and wearable sensors, act as physical sensing devices by reporting on-site information about objects and events of interest. Emerging examples include check-in applications, like Foursquare, and geotagging photo tools, like Flickr and Instagram, where individuals are actively providing fairly precise data about themselves, their ongoing and planned activities, as well as their current locations. A new paradigm has consequently emerged as the systematic activity of collecting and organizing geographic-related information, like descriptions of streets, buildings, and tourist spots. This paradigm, also called volunteered geographical information (VGI) or geospatial crowdsourcing, was defined in [9] as a process where citizens create and manage information, provided voluntarily, about the Earth and the environment.

In the advent of smart cities (and smart citizens), the acquisition of VGI data is being enabled by the increasing use of personal smart devices, which are equipped with sensors (e.g., Geographic Positioning System–GPS, proximity sensor, accelerometer, biometric sensor, and digital compass) capable of collecting accurately environmental measurements in real-time. The volume of this data is growing rapidly, particularly within the emergent fields of the Web 3.0 and *collective intelligence* [13]. Thanks to the Internet as well as the continuous progress in communication

✉ Michele Melchiori
michele.melchiori@unibs.it

Nafaâ Jabeur
nafaa.jabeur@gutech.edu.om

Roula Karam
roulakaramphd@gmail.com

Chiara Renso
chiara.renso@isti.cnr.it

[1] German University of Technology in Oman (GUtech), Athaibah, Oman

[2] Department of Information Engineering, Università degli Studi di Brescia, Brescia, Italy

[3] ISTI, CNR, Pisa, Italy

Published online: 09 March 2018

technologies, semantically rich user-generated geographical data are being widely shared and used by increasing numbers of applications across all fields, including health care, transportation, security, tourism, energy, and farming. A new range of applications are also being developed, where advanced intelligent mechanisms are deployed to combine the use of various types of data (e.g., structured data, unstructured data, real-time data, archive data, and open data) having unbalanced accuracies and multiple semantics. An example of such applications concerns the increasingly attracting topic of recommending itineraries for tourism activities.[1]

Recently, the interest in VGI has increased. For instance, compared to authoritative (i.e., official) geospatial data sources [9], which are maintained by governmental and for-profit organizations, VGI-based tools are particularly capable of delivering more timely, up-to-date, and detailed content. These benefits have spurred the development of large-scale initiatives, like the Open Street Map (OSM) and Wikimapia collaborative projects, for collecting, organizing, and developing volunteered geospatial data. For a better dissemination of this data, these projects have integrated capabilities from the social media (e.g., Flickr, Twitter, and Facebook) realm [15]. In spite of the considerable efforts being made to effectively use VGI, we argue that the underlying processes of content generation are intrinsically subjective and loosely controlled. Indeed, they often rely on untrained volunteers who are reporting data with different levels of precision from different devices. As a result, VGI content, which is typically heterogeneous in coverage, density, and quality, must be preprocessed in a convenient way to meet the requirements of its expected use [11]. To this end, effective techniques are still needed to assess and improve the quality of the user-generated data and ultimately ascertain their suitability for given domains or applications [12, 17]. Models and metrics for data quality (such as completeness, consistency, positional accuracy, temporal quality, and thematic accuracy) have been proposed and widely discussed within the context of assessing authoritative geospatial data [20]. Some approaches for assessing the quality of VGI data have been reviewed, classified, and discussed in [10]. In contrast with authoritative data, evaluating the quality of VGI is far more challenging [10]. In the literature, two types of approaches have been basically proposed to overcome these challenges. The first type of approaches consists in measuring the discrepancy between descriptions available in the VGI source being evaluated and descriptions available in a reference authoritative data source, assumed to be correct. To this end, the discrepancy is measured in terms of average completeness, accuracy, and coherence between the two sources over a designated spatial area [8]. In addition to being limited to an average evaluation of some parameters (i.e., completeness, accuracy, coherence, etc.), the performance of this first type of approaches depends highly on an authoritative reference source, which is not always available. The second type of approaches relies on defining indicators about aspects influencing the quality of data, without measuring this quality directly. These indicators include *lineage*, which considers the descriptions' history and evolution concerning, for example, a Point of Interest (PoI), *quality of textual descriptions* [3], *experience* [3], *trustworthiness*, and *reputation* [7].

In addition to measuring the quality of VGI descriptions, it is essential to estimate the reliability of each single VGI description. In other words, it could be critical to know whether a given description and its anonymous provider (VGI volunteer) are both trusted. In this case, measuring and assigning reputation scores to them will be valuable in deciding whether undesirable facts (e.g., contradictory, fake information, or ambiguous information) exist. As such, a measure of reputation is a helpful tool to make decisions, when either risk or uncertainty exists. These scores are of high importance, in several real-life scenarios using VGI, such as locating architectural barriers for people with disabilities [6], reporting on water well potability in developing areas [2], and monitoring industrial accidents [4].

In order to leverage the use of VGI, we are focusing in this paper on reputation as a quality indicator for both the VGI provider and its content. With respect to other works in the literature, our proposal aims to be more complete by providing a reputation model dealing with both time and changes in the described reality. Moreover, it provides a reference architecture, an algorithm, some experimental validation, and an applicative scenario in a single framework. The differences with other scientific proposals are discussed in "Section 2."

In particular, based on a multi-layer architecture, we have defined a comprehensive lightweight model of reputation in VGI that does not assume the existence of any specific metadata (such as rich users' profiles or social networks). Our model relies only on users' activities, which consist in creating/editing VGI descriptions and assigning feedbacks to existing descriptions created/updated by peers. This model exploits direct feedbacks expressed by users on VGI descriptions; however, it can work also without them, at the price of a lower accuracy. In addition, it is compliant with those VGI data sources that implement versioning mechanisms on data (i.e., maintaining the history of data versions over time for each piece of VGI content). Our contribution includes a novel algorithm to compute reputation scores. Experimentation of the proposed model in simulated scenarios showed that the reputation model performs generally with a good level of accuracy.

[1] https://get.google.com/trips/

The remainder of this paper is organized as follows. "Section 2" provides additional research context and motivations. In "Section 3," the multi-layer architecture is illustrated. In "Section 4," requirements for the reputation model, as well as for the metrics, are described. In "Section 5," experiments to assess the proposed solution are presented. In "Section 6," the applicative scenario about recommending tourist itineraries is presented to illustrate the potential advantages in adopting the proposed reputation mechanism. "Section 7" concludes the paper and points out some ideas for future works.

2 Related work

Research topics that are relevant to the contribution of this paper concern quality evaluation of VGI data sources, reputation/trustworthiness of georeferenced content, and its usage in crowdsourcing applications.

Quality in VGI datasources In their categorization of techniques for VGI improvement, Goodchild and Li [10] have classified the approaches that addressed the issue of assessing the quality of VGI data sources into three types: (1) those involving groups of people who check and correct individual contributions (crowdsourcing approaches), (2) those relying on trusted individuals who act as moderators (social-based approaches), and (3) those using domain specific knowledge and rules about geography. Other approaches, like in [8], aim to establish the average quality of VGI data on selected regions/areas of interest. To this end, they compare these data with authoritative datasets and measure the discrepancies between them in terms of standard quality metrics such as completeness, consistency, and positional/temporal accuracy. In a recent work, Touya et al. [19] have proposed a new approach that relies on some composable basic methods to assess the quality of crowdsourced point of interest (PoI) descriptions. Examples of these methods include (i) checking spatial relationships (e.g., the inclusion of a shop classified as PoI in the perimeter of a building); (ii) examining the sequence of editing steps applied to the description of a specific PoI; and (iii) measuring the similarity of crowdsourced descriptions with the corresponding descriptions obtained from an authoritative source. The proposed approach has been applied to some OSM data describing part of the city of Paris and revealed satisfactory results.

Reputation of georeferenced content Several research works have proposed techniques for estimating reputation of user-generated georeferenced content. For example, Bishr and Khun [2] have described a reputation model based on the consistency of the volunteers' reports concerning the potability of water wells in developing countries. The trustworthiness in a given report, about a well, is initially estimated based on its coherence with the majority of the previous reports on the same well. Therefore, the majority opinion is assumed as truthful. The trustworthiness in a report is reduced proportionally to the elapsed time since the creation of this report. The reputation score of a volunteer is adjusted, by increasing or decreasing it, based on the coherence of her/his reports with the majority opinion. Our model considers more general VGI scenarios allowing for complex descriptions of objects.

Zhao et al. [27] have proposed an approach where they estimate the trustworthiness level of a description A, concerning an object O, by considering the reputation of its author and, in case A updates a previous version A' describing the same object, the trustworthiness of A', too. In particular, the trustworthiness of A depends on (i) the author's reputation, (ii) the similarity distance between A and the previous version describing the same object, and (iii) the trustworthiness of the previous version. In addition to being inspired from the related work of Keßler et al. [14] and D'Antonio et al. [7], Zhao et al. [27] have proposed a detailed data schema supporting their model. Like in our approach, these authors distinguish between two ways of evaluating contributions: direct (i.e., by some user's feedback) and indirect (i.e., by some user's activities of editing VGI descriptions).

Reputation and trust in crowdsourcing applications Reputation and trust have been studied in various applicative domains, other than VGI. Both concepts have been always reported, in the literature, as interrelated. On the one hand, trust usually involves two subjects, A and B (e.g., user-user, user-service, agent-agent, etc.). Trust of A in B measures the degree of belief that A has about the intention and the capability of B to behave as expected [7, 22]. On the other hand, reputation is usually perceived as a collective (community-based concept) value of trustworthiness about the intention and capability of B to behave as expected [2, 22]. It is also a parameter being held by B as a means to attract other peers to communicate and coordinate with it. This parameter is obtained with respect to the truthful actions of B regarding other peers in the same environment [24]. Trust and reputation models have been particularly studied in multi-agent systems [26], social networks [18], and crowdsourcing systems [23]. In this latter field, which is the focus of our current work, assigning the right workers (or contributors) to the right tasks is still representing a critical and challenging issue. To deal with this issue, several approaches have proposed to maintain reputation/performance scores for workers and then recommend the most suitable of them (i.e., workers) for the list of available tasks accordingly. In this perspective, CrowdRec [23] has been proposed as

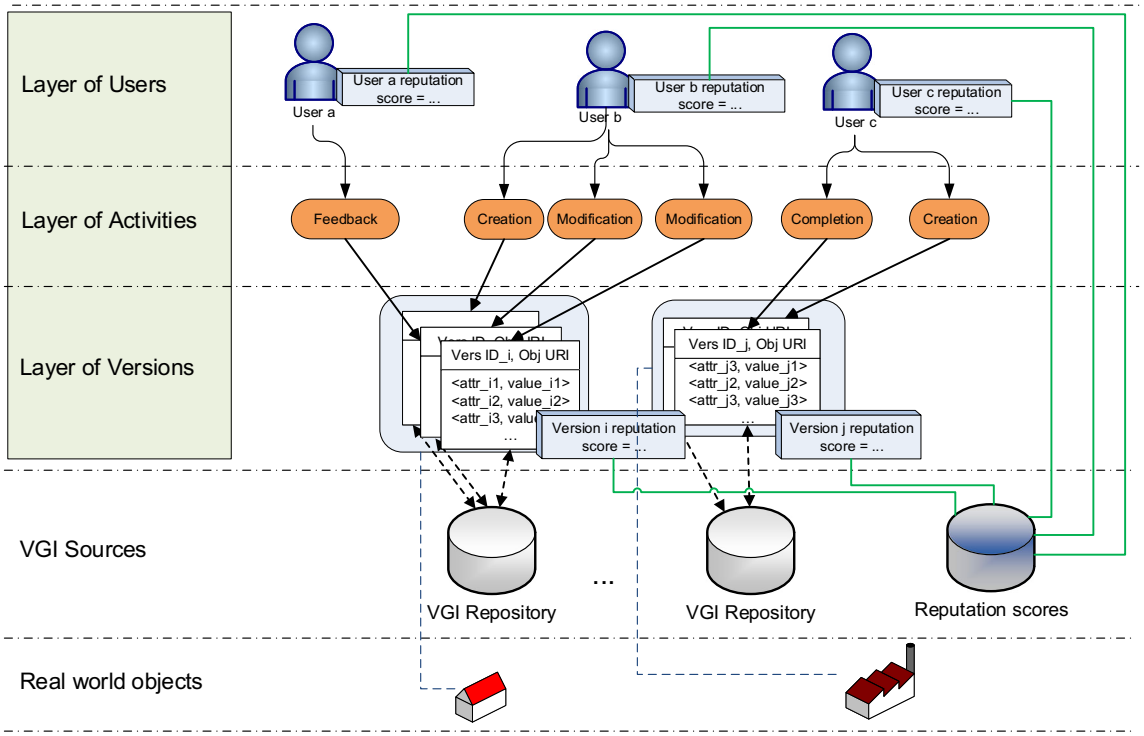

Fig. 1 Multi-layer reference architecture for reputation in VGI

a task-oriented recommendation approach to select reliable workers for a task assigned by a requester. A worker is recommended for a task of type T when s(he) has received an amount of positive evaluations on tasks of the same type T by trusted requesters. The approach defines a graph, where nodes represent requesters and edges represent trust relationships between them. Basically, an edge has a weight proportional to the number of times that the two requesters (i.e., the two nodes of the edge) have evaluated, in a coherent way, a worker on tasks of the same type. Like in our approach, CrowdRec deals with dishonest behaviors. However, the problem setting differs from ours since no overall reputation value is associated with a worker, but rather a *prediction of a worker's performance* on a specific type of task for a given requester is computed. SWORD [25] have addressed the allocation of tasks by permitting existing reputation models to be integrated in crowdsourcing systems. To this end, the authors have holistically considered the objectives of all stakeholders (including workers, system operators, and requesters) to express the allocation problem as a compromise between timeliness and quality. To reach satisfactory solutions for this problem, the authors have proposed a new constraint optimization-based approach with low computational complexity. Furthermore, Aroyo and Welty [1] have attempted to assess the trustworthiness of semantic annotations of textual contents when performed by different contributors. The authors have argued that some disagreement in users' annotations on the same text is unavoidable, particularly because of semantic ambiguity in the described object. Consequently, they considered an annotation to be acceptable when the disagreement does not exceed a given threshold. The authors have also proposed a new type of *gold standard/ground truth*, called CrowdTruth, that aims to model, realistically, the variety of human knowledge. The different perspectives and interpretations for each crowdsourced human annotation task are reflected in the disagreement between workers. In this case, the higher the disagreement of the worker with CrowdTruth, the lower the trustworthiness given to the worker, and vice versa.

3 Reputation-enhanced volunteered geographic information

We define the computation of reputation scores[2] in the context of a multi-layer reference architecture, as outlined in Fig. 1. The generation of VGI content is described in terms of users involved, activities performed, versions of VGI descriptions produced, and relationships among them. Computation of reputation scores focuses mainly on the layers of users, activities, and versions.

[2]Hereafter, we use the term *reputation score*, or simply *score*, in an interchangeable way to denote either the concept or its value, based on the context.

We first define the notion of a *version* for a VGI description (hereafter referred simply as *version*). According to [27], a version is a description produced by a user for a certain state of a real-world geospatial object, such as, for instance, the VGI description of a restaurant at the current time. A version is associated with the time instant (or time stamp) at which it is created. Several versions may exist for each real-world object, which are ordered by time stamp in a sequence referring to the same object. This is because the state of an object can change during its lifespan (e.g., the phone number or the name of a restaurant). In general, more than one version may have been produced even for the same object state (e.g., different users provide different information for the same restaurant). We recall that in our approach we maintain reputation scores for both users and versions. Basically, the score of a user depends on activities and feedbacks made by peers on any of the user's versions. Similarly, the reputation score of a version depends basically both on the feedbacks the version has received, and on the versions proposed at a later time for the same real-world object by other users.

In the *layer of versions*, the versions (of PoIs or spatial objects, such as roads) are modeled as sets of pairs $< attribute, value >$. A version is stored in a VGI repository, such as OSM and WikiMapia, and is associated with a creation time stamp (not depicted in Fig. 1) and a reputation score. Users modify versions by performing activities. The *layer of activities* describes these users' activities. In our current framework, we consider the following activities: (i) *creation* of a new version describing an object, (ii) *modification* on an existing version in order to propose a correction or updating, (iii) *completion* of an existing version in order to provide additional information, and (iv) *feedback* to express agreement or disagreement on a version. Each of these activities, except (iv), adds a version to a VGI repository.

Therefore, while the layer of versions describes the content of VGI data sources in terms of versions, the reputation scores of users and versions are maintained in a dedicated storage, separated from the original VGI repositories (see VGI Sources, Fig. 1). This allows us to process reputation scores independently of pre-existing VGI repositories, which are not required to be modified. Users are explicitly described in the *layer of users* and are distinguished based on their past behavior. In fact, we differentiate *active* users, who have performed some sort of activity, from *inactive* users, who either have not performed any activity at all or have given feedbacks only. We exemplify these two types of users in Fig. 1, where *User b* and *User c* are active, whereas *User a* is inactive, since s(he) has produced feedbacks only. Concerning updating of scores, an activity of modification, completion, or feedback on a version triggers an updates of the scores for both the version and its author. User activities and their effects on reputation scores are summarized in Fig. 2, before being formalized in detail in "Section 4."

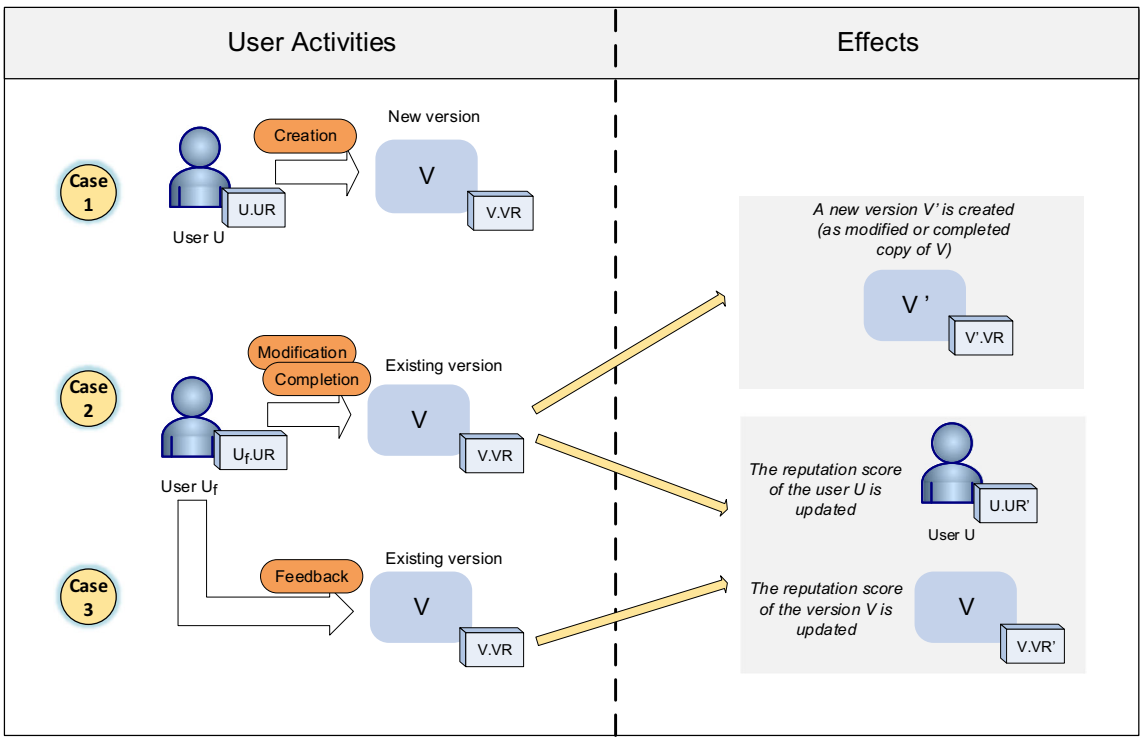

Fig. 2 User activities on versions and their effects on reputation scores

A user U produces a new version V (Case 1 in Fig. 2) as a set of pairs $< attribute, value >$ describing the state of a real-world object. This version is assigned with an initial reputation score, denoted as $V.VR$, which depends on the author's reputation score, denoted as $U.UR$.

In case a modification activity is performed on an existing version V, the user U_f produces a new version V' as a copy of V, but with modified values and/or attributes for some of the original pairs of V. Modification also includes deleting pairs. For example, s(he) can modify the pair <open, Monday-Friday> to <open, Monday-Saturday> or <beginningAt, 8:00> to <openingAt, 8:00>. The effects on reputation scores for a modification on V (Case 2 in Fig. 2) performed by the user U_f consist in updating (i) the reputation score $V.VR$ of V, and (ii) the reputation score $U.UR$ of U, author of V. The updated scores are denoted as $V.VR'$ and $U.UR'$, respectively.

The case of a completion activity (still Case 2 in Fig. 2) is similar to the previous one. A completion activity consists in creating a new version V' as a copy of V, then extending this copy with additional content (e.g. new pairs). For example, the pair <phone2, +39303333020> is added. Concerning the effects of a completion activity, both (i) the reputation score of U, author of the existing version V, and (ii) the reputation score of the version V, are updated, as it happens in the case of a modification activity. Finally, a feedback expressed by a user U_f on an existing version V (Case 3 in Fig. 2) aims to assess the correctness of this version. A set of discrete values could be used to reflect degrees of correctness ranging from totally correct (extremely positive feedback) to totally incorrect (extremely negative feedback). As effects of the feedback activity, both (i) the reputation score $U.UR$ of the user U (author of the version V) and (ii) the reputation score $V.VR$ of the version V are updated as $U.UR'$ and $V.VR'$, respectively.

4 Reputation evaluation model

In this section, we propose requirements and metrics that are based on the multi-layer architecture outlined in Fig. 1. Then, we provide an algorithm that implements our reputation model. As mentioned above, reputation of versions depends on several criteria, including feedbacks (also called direct evaluations) as well as activities of modification and completion (also called indirect evaluations). In particular, a modification on a version is considered as an indirect negative feedback. It is expected to reduce the reputation of the original version in accordance with the following rationale: the user producing the modified version has judged the original version as presenting some errors or requiring updating. On the contrary, a completion tends to recognize the content of an existing version as correct even though incomplete and it extends the content of the original version. Therefore, a completion is considered as an indirect positive feedback and it is expected to increase the reputation of the original version. Moreover, the impact of a modification/completion activity on reputation has to be proportional to the amount of proposed changes with respect to the original version. For example, in case of modification, a small amount of changes should reduce the reputation of the original version only slightly. On the other hand, a large amount means that the original version was recognized as largely incorrect/incomplete and this should decrease its reputation to a larger extent.

To ensure consistency in user-produced content, as well as to prevent malicious users from trying to increase or decrease reputation scores deliberately, the activities permitted on versions are subject to restrictions. For instance, a user is allowed neither giving directly a feedback to another user nor giving more than one feedback to a version. This limits the opportunities for a malicious user to increasing or decreasing reputation scores of specific peers, or of their contents, by expressing multiple feedbacks deliberately.

In the following, we define the metrics that formalize the reputation scores. To this end, we will describe how the reputation scores of versions and users are generated and updated.

Reputation of versions The reputation score of a version v, denoted as $v.VR$, is based on direct and indirect evaluations (feedbacks and activities of modification/completion) performed on v by users different from its author. The score $v.VR$, like the other reputation scores, falls within the range [0, 1] and is calculated according to the following equation:

$$v.VR = \begin{cases} v.VR_0 \\ (1 - e^{-k}) \cdot \frac{v.POS}{v.POS + v.NEG} + e^{-k} \cdot v.VR_0 \end{cases} \quad (1)$$

where, $v.VR \in [0, 1]$ and k is the total number of evaluations on v. The first case in (1) applies when $k = 0$. In the second case, when $k \neq 0$, the reputation score $v.VR$ is defined based on the values $v.POS$ and $v.NEG$, representing the cumulative values of the positive and negative evaluations on the version v, respectively. These values will be defined shortly in this section. The role of the exponential coefficients, e^{-n} and $(1 - e^{-n})$, is to reduce the amplitude of the initial oscillations in the $v.VR$ value when k is low (e.g., $k < 5$), thereby giving more importance to the initial reputation VR_0. This prevents a single positive or negative evaluation from either increasing or decreasing a the reputation $v.VR$ considerably when v has received just few evaluations. In fact, as k grows, the impact of the initial reputation $v.VR_0$ on the reputation score lessens, as, in this case, the latter will increasingly depend on the evaluations expressed by other users (i.e., on the values of $v.POS$ and $v.NEG$).

Initialization of the version reputation score The initial reputation score $v.VR_0 \in [0, 1]$ of a version depends on the reputation of its author u as well as on the activity that led to this version (i.e., creation, modification, and completion). It is calculated as follows:

$$v.VR_0 = \begin{cases} u.UR & \text{if } new(v) \\ (1 - Sim(v, vPrev)) \cdot u.UR + \\ Sim(v, vPrev) \cdot min(vPrev.VR, u.UR) & \text{otherwise} \end{cases} \quad (2)$$

where, predicate $new(v)$ is true when v is a completely new version (i.e., the result of a creation activity). In this case, the initial reputation $v.VR_0$ of v is set equal to the reputation $u.UR$ of its author. In case v is obtained by modification or completion of an existing version $vPrev$, $v.VR_0$ is computed as a weighted mean of two values, namely, (i) the reputation $u.UR$ of the activity performer and (ii) the minimum between the reputation of the version $vPrev$ and $u.UR$. The rationale for selecting the minimum is the following. When the reputation score of $vPrev$ is high and the reputation score $u.UR$ is low, a malicious user u could modify/correct a version with a high reputation in order to produce a modified version v of it with some spurious content (e.g., providing a different website address), but having initially a high reputation score. On the contrary, this cannot happen if we take the minimum between the values $vPrev$ and $u.UR$, because the user's reputation $u.UR$ sets an upper bound to the initial reputation score of the version v.

Moreover, the term $Sim(v, vPrev) \in [0, 1]$ is a measure of the similarity between the current and the previous modified/corrected version. Its value ranges from 1, when versions are the same, to 0 when the versions do not share any common content. The more dissimilar these versions (i.e., $Sim(v, vPrev)$ is closed to 0), the closer the version reputation score to the author's reputation score $u.UR$ and thus independent of the reputation of $vPrev$. This is done to cope with the scenario in which, when the version v modifies/completes $vPrev$ with a significant amount of new content, the reputation of v should depend more on the author's reputation than on the reputation of the original version $vPrev$.

Example In order to clarify this idea, let us consider a case where the original version $vPrev$ and an updated version v of it, proposed by a user with low reputation (e.g., $u.UR = 0.1$), are quite different (let us suppose $Sim(vPrev, V) = 0.2$).

According to (2), the initial reputation is $v.VR_0 = (1 - 0.2) \cdot u.UR + 0.2 \cdot min(vPrev.VR, u.UR) = 0.8 \cdot 0.1 + 0.2 \cdot min(vPrev.VR, 0.1)$, that is, a value close to 0.1 since $vPrev.VR \in [0, 1]$. Hence, as desired, the value $v.VR_0$ is similar to the value of the author's reputation score.

If the original and the updated versions are very similar (e.g., $Sim(vPrev, v) = 0.9$) and the author of the updated version has low reputation, e.g., $u.UR = 0.1$, then the reputation of the updated version is bounded. This upper bound is the minimum between the author's reputation and the original version reputation (e.g., 0.1). Again, the model is coherent with the rationale that an author with low reputation score cannot produce a version with an initial high reputation score.

The details on how computing $Sim(v, vPrev)$ are not provided here because the definition of similarity measures is a topic discussed widely in the literature (see, for example, [16]). Nevertheless, we underline that $Sim(v, vPrev)$ is proportional to the number of pairs $< attribute, value >$ shared by both of the versions v and $vPrev$.

Feedback evaluation The values $v.POS$ and $v.NEG$, summarizing positive and negative effects of evaluations (feedbacks and indirect feedbacks) on a version v are updated every time an evaluation on v is produced. The impact of an evaluation made by a user u_f is proportional to the reputation of u_f. In particular, the equation for updating the current $v.POS$, when either a positive feedback or a completion f is submitted by user u_f having reputation $u_f.UR$, is the following:

$$v.POS' = v.POS + \begin{cases} u_f.UR & \text{if } f \text{ is pos. feedback} \\ Sim(v', v) \cdot u_f.UR & \text{if } f \text{ is completion} \end{cases} \quad (3)$$

In case of a completion, we denote with v' the upgraded version of v resulting from the completion activity. The more dissimilar the versions v and v' (i.e., $Sim(v', v) \approx 0$), the smaller the increase of $v.POS$. This is due to the similarity coefficient $Sim(v', v)$. The rationale is that, when v' is very different from v, this is not considered a completely positive evaluation of v, i.e., v was evaluated as rather incomplete. In this case, $v.POS'$ should not differ from $v.POS$ consistently. The difference between $v.POS$ after and before its most recent updating is denoted as $\Delta v.POS$ (i.e., $\Delta v.POS = v.POS' - v.POS$).

The equation for updating $v.NEG$, when either a negative feedback or a modification f is submitted by u_f with reputation $u_f.UR$, is the following:

$$v.NEG' = v.NEG + \beta \cdot \begin{cases} u_f.UR & \text{if } f \text{ is neg. feedback} \\ (1 - Sim(v', v)) \cdot u_f.UR & \text{if } f \text{ is modification} \end{cases} \quad (4)$$

The more dissimilar the versions v and v', the higher the increase of $v.NEG$, owing to the term $(1 - Sim(v', v))$.

This accounts for the fact that a large modification is a clear negative evaluation of the version v. The difference between $v.NEG$ after and before its most recent updating is denoted as $\Delta v.NEG$ (i.e., $\Delta v.NEG = v.NEG' - v.NEG$). The role of the coefficient β is discussed hereafter.

Catching status changes in objects The coefficient β, having a value ≥ 1, permits to increase the effect of a negative evaluation when a specific situation occurs, which provides an indication of a (possible) change in the real-world object described by v. Specifically, if the object is described correctly by v, then it is likely that v has a high reputation owing to the evaluations received (e.g., we have $v.POS > v.NEG$). If, later on, a change in the status of the object occurs (e.g., some feature changes), then v is no more representing a correct description of it and we expect users to start producing evaluations of v that are prevalently negative. Because v is no longer correct, its reputation should decrease quickly, in spite of its past high reputation. A way to obtain this behavior in (4) is to overweight the effect of the negative feedbacks received after the change of status, by augmenting the value of β. Formally, β is defined as follows:

$$\beta = \begin{cases} \gamma & \text{if } v.POS > v.NEG \text{ and} \\ & \quad lastN(k, v) > lastP(k, v) \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where, $lastN(k, v)$ and $lastP(k, v)$ are functions returning the number of negative and positive evaluations, respectively, among the last k evaluations concerning the version v. The parameter k is a positive and odd integer. It is odd because the purpose is to establish whether the majority of the k evaluations is negative or positive. In our experiments, we set $k = 7$. For example, if a version v has high reputation score (e.g., $v.POS > v.NEG$) but it has received two positive evaluations and five negative evaluations as the latest seven ones (e.g., $lastN(7, v) > lastP(7, v)$), this is meant as an indication of a change of status in the described real-world object. Consequently, according to (5), β is set to the value γ. This modifies the way $v.NEG'$ is updated by (4). The constant value $\gamma > 1$ is therefore the coefficient used to increase the significance of a negative evaluation in (4), as explained above.

The actual value γ is the result of a tradeoff and, after performing various simulations, where we varied it and evaluated the obtained results, we set $\gamma = 3$. On the one hand, when the first case in (5) holds, this reveals a possible, but not sure, change of status. Hence, setting γ with a value too high would reduce the reputation of v too quickly when few close negative evaluations are received. On the other hand, a value for γ too low has the drawback that it minimizes the ability of the model to detect changes of status.

User reputation Every time an activity, as a feedback, a completion, or a modification, is performed on a version, the *user reputation* score of its author u is updated as follows:

$$u.UR = \begin{cases} UR_0 \\ (1 - e^{-n}) \cdot \frac{u.POS}{u.POS + u.NEG} + e^{-n} \cdot UR_0 \end{cases} \tag{6}$$

where $u.UR \in [0, 1]$, with $u.UR = 1$ being the maximum reputation. It is worth to say that our model is not classifying the honest/malicious nature of users sharply, but all the user's activities are weighted proportionally to her/his reputation. The first case of (6) applies when no evaluation has been made so far by other users on any of the versions authored by the user u; in other words, the number of positive and negative feedbacks received is 0 and therefore $u.POS$ and $u.NEG$ are 0. The term UR_0 is the initial user reputation, and a possible strategy is to set it to a low reputation value (e.g., 0.3) for each new user. Alternatively, the initial reputation could be set based on the automated analysis of information on the user profile (e.g., its completeness) provided when filling in the registration profile.[3] The second case of (6) applies when at least one evaluation has been performed and is defined according to an expression similar to the one for reputation of versions, as in (1). The parameter n is the total number of versions produced by the user u. This number accounts for new versions, obtained by creation activities, and those produced by either modification or completion of existing versions.

The terms $u.POS$ and $u.NEG$ represent the positive and negative cumulative evaluations, respectively, made by other users of all the versions authored by the user u. These terms are initially set to 0. When a version v, created by the user u, receives a positive evaluation, then $u.POS$ is updated to a new value $u.POS'$ according to

$$u.POS' = u.POS + \Delta v.POS \cdot h(t, t_v) \tag{7}$$

The expression for updating $u.NEG$, based on a negative evaluation on v, is defined in a similar way:

$$u.NEG' = u.NEG + \Delta v.NEG \cdot h(t, t_v) \tag{8}$$

where $ht(t, t_v)$ is a coefficient weighting the contribution of $\Delta v.POS$ and $\Delta v.NEG$ to $u.POS'$ and $u.NEG'$, respectively, based on the age of the version. The term t_v is the creation time of the version v and t is the time the evaluation is received (i.e., the evaluation time). The expression of $ht(t, t_v)$ is formalized below.

[3] The features in user profile can also be analyzed by machine learning techniques in order to detect fake profiles or even multiple registrations of the same user, as proposed in Xiao et al. [21]. Then, fake profiles can be disabled or submitted to human checking.

Algorithm 1 Reputation evaluation algorithm

Input : the sequence Σ of events: users' registrations and activities; the initial user reputation value UR_0; the period α; the parameters k and γ for change detection.

Result: updated reputation scores of users and versions.

while *True* **do**
  get next $event(t_i)$ in Σ;
  switch $event(t_i)$ **do**
    case new user u has registered
      $u.UR \leftarrow UR_0$;
      break;
    case new version v is created by user u
      $v.VR \leftarrow u.UR$;
      break;
    case v' obtained by modification of v, by user u_f
      $s \leftarrow Sim(v', v)$;
      /* set the reputation of v' using the second case of (2) */
      $v'R_v \leftarrow (1-s)\cdot u_f.UR + s \cdot min(v.VR, u_f.UR)$;
      /* update $v.NEG$ using the 2nd case of (4) */
      $\beta \leftarrow$ BETAEVAL(Σ, $v.POS$, $v.NEG$, v, γ, k);
      $v.NEG_{OLD} \leftarrow v.NEG$;
      $v.NEG \leftarrow v.NEG + \beta \cdot (1-s) \cdot u_f.UR$;
      /* update reputation of u author of v */
      $u.NEG \leftarrow u.NEG + (v.NEG\text{-}v.NEG_{OLD}) \cdot h(t, t_i)$;
      $u.UR \leftarrow$ UPDUSERREP(u, UR_0, $u.POS$, $u.NEG$);
      break;
    case v' obtained by completion of v, by user u_f
      $s \leftarrow Sim(v', v)$;
      /* set the reputation of v' using the second case of (2) */
      $v'R_v \leftarrow (1-s)\cdot u_f.UR + s \cdot min(v.VR, u_f.UR)$;
      /* update $v.POS$ using the 2nd case of (3) */
      $v.POS_{OLD} \leftarrow v.POS$;
      $v.POS \leftarrow v.POS + s \cdot u_f.UR$;
      /* update reputation of u author of v */
      $u.POS \leftarrow u.POS + (v.POS\text{-}v.POS_{OLD}) \cdot h(t, t_i)$;
      $u.UR \leftarrow$ UPDUSERREP(u, UR_0, $u.POS$, $u.NEG$);
      break;
    case positive feedback on v, by user u_f
      /* update $v.POS$ using the 1st case of (3) */
      $v.POS_{OLD} \leftarrow v.POS$;
      $v.POS \leftarrow v.POS + u_f.UR$;
      /* update reputation of u, author of version v */
      $u.POS \leftarrow u.POS + (v.POS\text{-}v.POS_{OLD}) \cdot h(t, t_i)$;
      $u.UR \leftarrow$ UPDUSERREP(u, UR_0, $u.POS$, $u.NEG$);
      break;
    case negative feedback on v, by user u_f
      /* update $v.NEG$ using the 1st case of (4) */
      $\beta \leftarrow$ BETAEVAL(Σ, $v.POS$, $v.NEG$, v, γ, k);
      $v.NEG_{OLD} \leftarrow v.NEG$;
      $v.NEG \leftarrow v.NEG + \beta \cdot u_f.UR$;
      /* update reputation of u, author of version v */
      $u.NEG \leftarrow u.NEG + (v.NEG\text{-}v.NEG_{OLD}) \cdot h(t, t_i)$;
      $u.UR \leftarrow$ UPDUSERREP(u, UR_0,$u.POS$, $u.NEG$);
      break;

Algorithm 2 User reputation updating

```
1 Function UpdUserRep(u, UR_0, u.POS, u.NEG): float is
2     /* update the user reputation by using the second case of (6) */
3     n ← u.number_of_versions;
4     UpdatedRep ← (1 − e^n) · u.POS/(u.POS + u.NEG) + e^n · UR_0;
5     return UpdatedRep;
6 end
```

Algorithm 3 Evaluation of the β coefficient

```
1  Function BetaEval(Σ, v.POS, v.NEG, v, γ, k): float is
2      /* compute the beta coefficient by using (5) */
3      β ← 1;
4      if (v.POS > v.NEG) then
5          lastN(k, v) ← number of negative feedbacks among the last k events of Σ;
6          lastN(k, v) ← lastN(k, v) + number of modifications to v among the last k events of Σ;
7          lastP(k, v) ← number of positive feedbacks among the last k events of Σ;
8          lastP(k, v) ← lastP(k, v) + number of completions to v among the last k events of Σ;
9          if (lastN(k, v) > lastP(k, v)) then
10             β ← γ;
11         end if
12     end if
13     return β;
14 end
```

Aging of versions The coefficient $h()$, appearing in (7) and (8), assigns a higher weight to evaluations expressed on versions produced recently. This weight decreases linearly since the difference between the time t—when the evaluation on a version v is given—and the time t_v—when the version v was created—is larger. In particular, $h(t, t_v)$ is equal to 0 when this difference is greater than a predefined value α. The rationale is that an evaluation on an old version should impact less or nothing the author's reputation compared with an evaluation on a newer version because version content can become out-of-date as time passes. When a version is out-of-date, it is more prone to receiving negative feedbacks, independently of its quality at the creation time. In our experiments, we have chosen α equal to 180 days; therefore, feedbacks on a version created more than 180 days before do not change the reputation of the version author. This coefficient is defined as

$$h(t, t_v) = \begin{cases} \frac{\alpha-(t-t_v)}{\alpha} & \text{if } t - t_v < \alpha \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where, $h(t, t_v) \in [0, 1]$ and its value is 1 when $t = t_v$. Notice that $u.POS$ and $u.NEG$ are always non-negative.

4.1 Activity-driven procedure for reputation updating

The metrics presented above permit to define an algorithm for computing reputation scores. In particular, Algorithm 1 shows in an operational style how these metrics are combined to process user activities, as defined in "Section 3." Occurrence of user activities is represented as a sequence of events and the body of the algorithm is a loop where (i) an $event(t_i)$ (row 2) corresponding to some user's activity performed at time t_i is received and (ii) operations associated with a specific case, selected based on the activity type (rows from 3 to 49), are performed. The result is an updating of reputation scores for both users and versions depending on the activity represented by $event(t_i)$. The main algorithm invokes two additional functions described in Algorithm 2 and in Algorithm 3. The first one implements the updating of user reputation scores as defined by the second case of (6). The second function, based on (5), implements the evaluation of the coefficient β devoted to the detection of possible status changes in real-world object, as discussed previously.

5 Experimental evaluation

This section provides experimental results to evaluate (1) the behavior and performance of our approach and (2) to observe how some factors influence performance.

5.1 Experimental setup

The reputation model described in the previous section has been implemented in Matlab[4]. Simulations have been performed to evaluate the model accuracy in two different experimental settings.

In this implementation, versions and users are described as tuples. The components of these tuples are pairs <attribute, value>. In particular, a version is a tuple with attributes *identifier*, *author identifier*, *creation time*, current and previous

[4] https://www.mathworks.com/products/matlab.html

values of POS and NEG, *number of received feedbacks*, (current) *reputation score*, *initial reputation score*, *correctness* (i.e., the version is a priori correct or not), and *status* (i.e., valid or obsolete, w.r.t. the described object). Likewise, a user in the context of the simulations is a tuple with the attributes *user identifier*, *initial reputation score*, *reputation score*, *cooperative status* (i.e., the user is cooperative or not), *active status*, *number of performed activities*, and values of POS and NEG.

Each version is randomly labeled either as correct (i.e., supposed to represent faithfully some real world object) or incorrect at the beginning of the simulation by setting the *correctness* attribute value. Similarly, after its creation, a user is randomly labeled either as cooperative (i.e., behaving honestly) or non-cooperative (i.e., behaving maliciously) by setting the value of *cooperative status*. The probabilities for setting as correct/incorrect versions and as cooperative/non-cooperative users are the (input) parameters of the simulation. Concerning activities, cooperative users can (i) create correct versions and (ii) perform activities on existing versions coherent with their status, e.g., giving positive feedbacks to correct versions and making modifications on incorrect versions. We recall that a modification is an indirect negative evaluation. Non-cooperative users can (i) create incorrect versions, (ii) perform activities on existing versions coherent with their status, e.g., giving negative feedbacks, or making modifications, to correct versions. Moreover, users are established a priori as either active or inactive by setting the value of the attribute *active status*. We recall that active users can perform any type of activity on versions while inactive users only express feedbacks. The percentage of active users is a simulation parameter, which is set by default to 10%; hence, 90% of users are inactive. This proportion between active users, contributing as active reviewers of the VGI content, and inactive users, giving only feedbacks, is expected to be quite realistic because it reflects the proportion known for members of the Amazon community.[5]

Basically, in the simulations, at each step a user is chosen randomly to perform some activity on some version. Both the type of activity and the version are chosen randomly, too. Consequently, the Matlab implementation of the Algorithm 1 computes and updates the reputation scores based on the activity performed. A simulation terminates after a number of activities given has been executed. This number is a parameter fixed at the start of the simulations.

The type of each activity performed by a user is chosen randomly according to a probability distribution given. In the experiments, we used the following one: *creation of a new version*, 1%; *modification*, 1%; *completion*, 1%; *feedback*, 97%. We assume this distribution is reasonable since it is coherent with a real setting in which most of the users provide just feedbacks because of the limited effort required. On the contrary, creation, modification, and completion are typically far less frequent because of the bigger effort and time required to produce/modify the content. We set the total number of activities to a high value (i.e., 10,000), and the number of users to 500, in order to observe the model behavior asymptotically. The number of activities performed per day is also a parameter of the simulation (we set it to 50 in our experiments). We expect that, if the majority of users are cooperative and the majority of versions are correct, then a low number of activities and users deliver in general less good results, while higher numbers of users and activities give better results. The default values for the simulation parameters are reported in Table 1.

The main objective of the experiments is to measure the accuracy of the reputation model in identifying correctly, i.e., according to the initial assignment, users as cooperative/non-cooperative, and versions as correct/incorrect.

We recall that the proposed reputation model tries to assign high reputation scores to cooperative users and to correct versions and assigning low reputation scores otherwise. In the experiments, we assume that a cooperative user is identified correctly when the reputation score is in the range $(0.5, 1]$ at the end of the simulation; a non-cooperative user is identified correctly when the reputation score is in the range $[0, 0.5]$. We use the same criteria for the identification of versions.

We experimented two main scenarios in order to observe the level of accuracy under different conditions, as illustrated

Table 1 Default parameters of the performed simulations

Parameter	Value
Number of activities performed in the simul.	10,000
Number of users	500
Percent of active users	10%
Activities per day	50
Probability distribution for user activities	Creation, 1%
	Modification, 1%
	Completion, 1%
	Feedback, 97%
Initial reputation UR_0	0.3
Aging parameter α	180 days
Probability of status change for objects	1% per day

[5] https://www.quora.com/What-percentage-of-buyers-write-reviews-on-Amazon

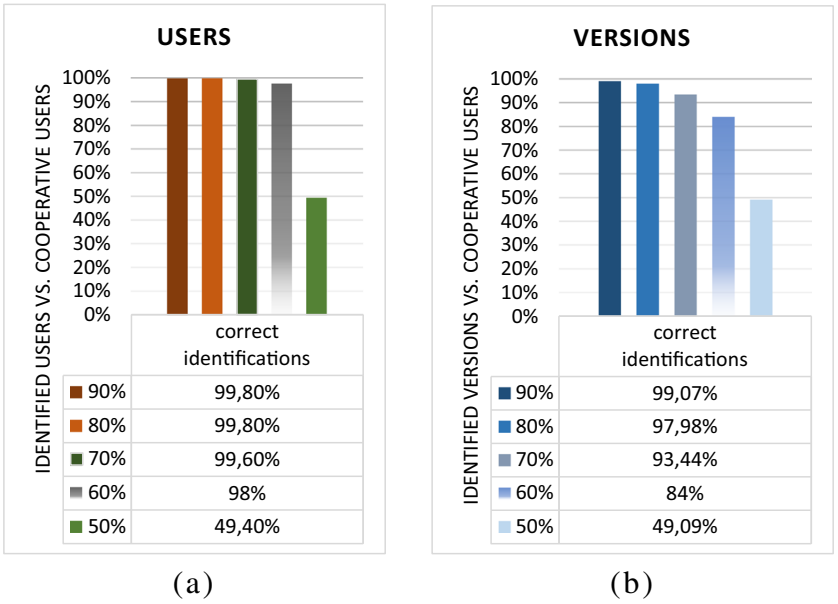

Fig. 3 Identification rates vs. cooperative user rates (simple scenario), for **a** users and **b** versions

in the following two sections. All the numerical results presented are obtained as average outputs of ten simulations.

5.2 Evaluating model accuracy with no aging of versions

The objective of this first experiment is to compute the accuracy of identifying cooperative and non-cooperative users as the percent of cooperative users is varying. Here, we focus on a simple scenario where objects/PoIs described by versions do not change their status during the simulation. Consequently, a version describing an object correctly is considered to maintain its correctness for the whole simulation. In this scenario, therefore, we disregard the aging of versions features (i.e., we set $h(t, t_v) = 1$) to observe the level of accuracy in a simple scenario. The results reporting the amount of users identified correctly at the end of the simulations are shown in Fig. 3a. The results reporting the amount of versions identified correctly are shown in Fig. 3b.

We can notice the high identification rates of the model when the cooperative users are 90%, 80%, 70%, and 60%, respectively, of the total users. Results were even better when we considered the amount of correct identifications of versions that received at least five feedbacks (not reported in Fig. 3). This confirms that the model, as it is based on user activities, performs better when most of the users are cooperative. On the contrary, when the amount of

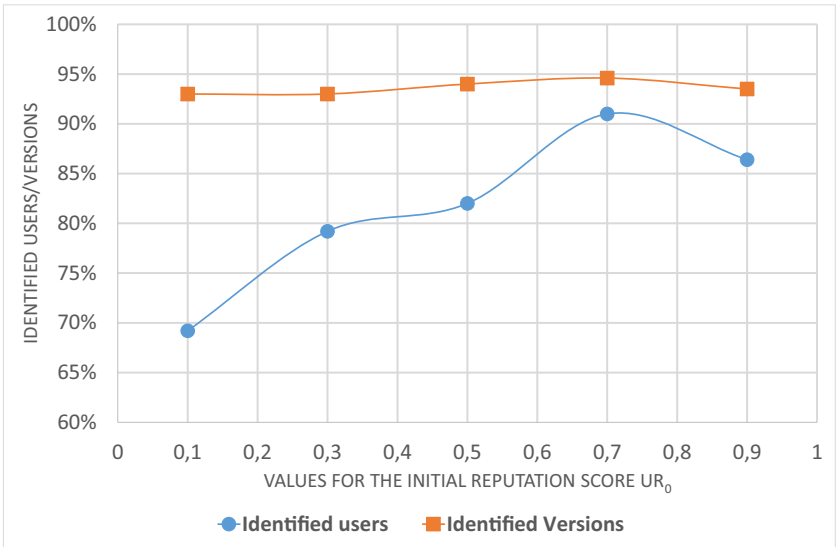

Fig. 4 Identification rates for different values of the initial reputation UR_0 after performing 2000 activities, with 70% of cooperative users

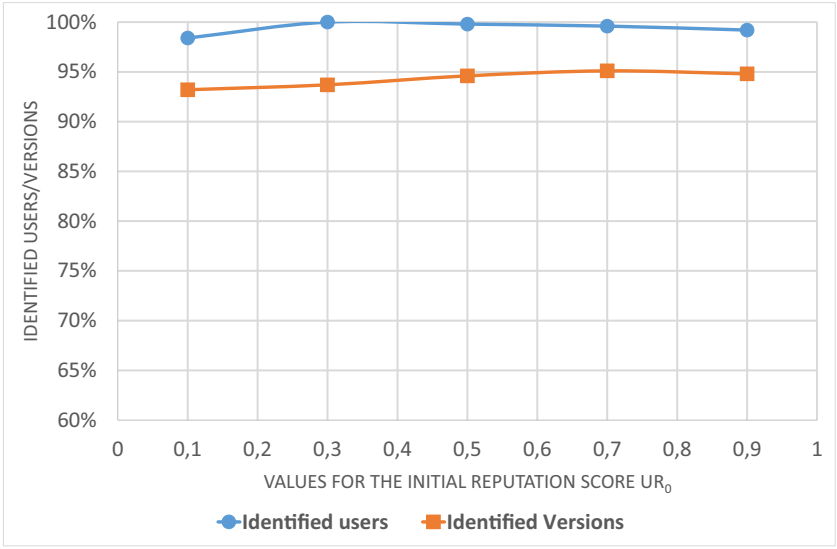

Fig. 5 Identification rates for different values of the initial reputation UR_0 after performing 10,000 activities, with 70% of cooperative users

cooperative users decreases to 50%, we notice how the identification rate falls down. In this case, since cooperative and non-cooperative users are equally balanced and every user is assigned the same initial reputation score UR_0, the model is clearly not effective in identifying users as cooperative or not. Concerning the initial user reputation parameter UR_0, a set of experiments have been performed by varying its value. In the first experiment, we run simulations, each one producing 2000 activities and using a different UR_0 value (i.e., 0.1, 0.3, 0.5, 0.7, 0.9), with 70% of cooperative users. The identification rates obtained for users and versions are shown in Fig. 4. The highest rates correspond to $UR_0 = 0.7$. So, the choice of value of UR_0 has an effect which is relevant to the model accuracy when a quite low number of activities is performed, as in this case. In a second experiment, the number of activities has been set to a higher value (i.e., 10,000). The identification rates are shown in Fig. 5. In this case, we observe how the different values for UR_0 have a low effect on the results, which are similar.

5.3 Evaluating model accuracy with aging of versions

In the second set of experiments, we simulate the model including the aging of versions. In this scenario, (i) reputation scores are updated keeping into account the aging of versions by means of the coefficient $h(t, t_v)$ defined in

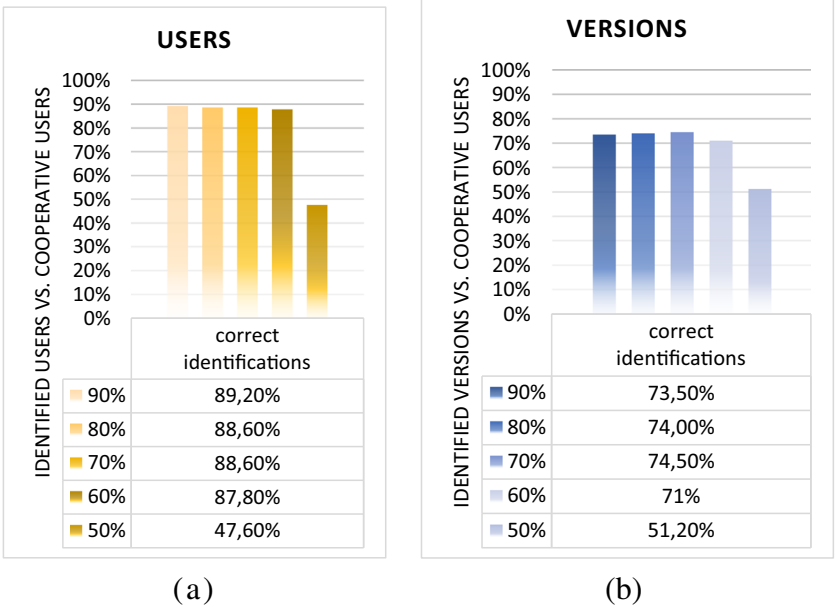

Fig. 6 Identification rates vs. cooperative users rates (complex scenario, with aging of versions and changes of status), for: **a** users, **b** versions

Table 2 Comparison of identification rates with and without versions aging, for different amounts of cooperative users, after performing 10,000 activities

Aging function	Objects	90%	80%	70%	60%
With $h() = 1$	Users	81.0%	81.4%	80.0%	81.0%
	Versions	70.0%	74.0%	73.3%	71.0%
With $h()$ as in (9)	Users	89.2%	88.6%	88.6%	87.8%
	Versions	73.5%	74.0%	74.5%	71.0%

(9); and (ii) during the simulation. a correct version becomes incorrect according to a given probability, which is listed in Table 1 as "Probability of status change for objects."

In particular, aging of versions is a mechanism relevant for objects that can change their status (e.g., a PoI whose phone number may change frequently) while it is less relevant for objects with more static features (e.g., a monument). In the first case, it is desirable that negative feedbacks and modifications on a version older than a predefined α period do not reduce the reputation of the author of this version. In the following experiments, α is set to 180 days, but it is a parameter of the model and may change according to the specific domain of the type of PoI. For example, for historical monuments, the content of versions does not tend to change frequently, while in dynamic environments, like shops and bars, information tends to change more often. The rationale for using in the simulation quite a number of days, like this, is because we observe that it is quite common for real-world objects to change at least one of their descriptive features after this time passed.

The rates of correct identifications are shown in Fig. 6a for users and in Fig. 6b for versions. We notice how the performance is lower compared to the values obtained in the previous simpler scenario. However, it shows how the model keeps good efficiency also in this scenario.

Evaluating the aging mechanism. The question now becomes: To what extent is the mechanism of aging of versions (i.e., the $h()$ expression) effective in this scenario? In the next experiments, a version initially correct can become incorrect with a certain probability during the simulation. Then, we compare the identification rates in the case $h(t, t_v)$ set to 1 and in the case $h(t, t_v)$ defined as in (9). This is repeated for different amounts of cooperative users. The results, reported in Table 2, seem to corroborate the positive contribution to the identification rates of the aging of versions introduced by (9).

Evaluating different distributions of activities. The previous experiments used a default distribution of user activities, which is the one reported in Table 1. We can question how much the model accuracy is affected by the activity distribution. In particular, we made experiments with two additional distributions: (a) *creation of a new version*, 1%; *modification*, 5%; *completion*, 5%; *feedback*, 89%; and (b) *creation of a new version*, 5%; *modification*, 5%; *completion*, 5%; *feedback*, 85%. In this case, the number of activities performed is 10,000 with 70% of cooperative users. The results reported in Fig. 7, for the default and for the two additional distributions, seem to indicate that a lower amount of feedbacks (i.e., 89% and 85%, w.r.t. 97% of the default distribution) reduces the accuracy in identifying versions correctly. Therefore, in our model, direct evaluations (i.e., feedbacks) have a higher effect on determining correctly the reputation scores of versions than indirect evaluations (i.e., modifications and completions).

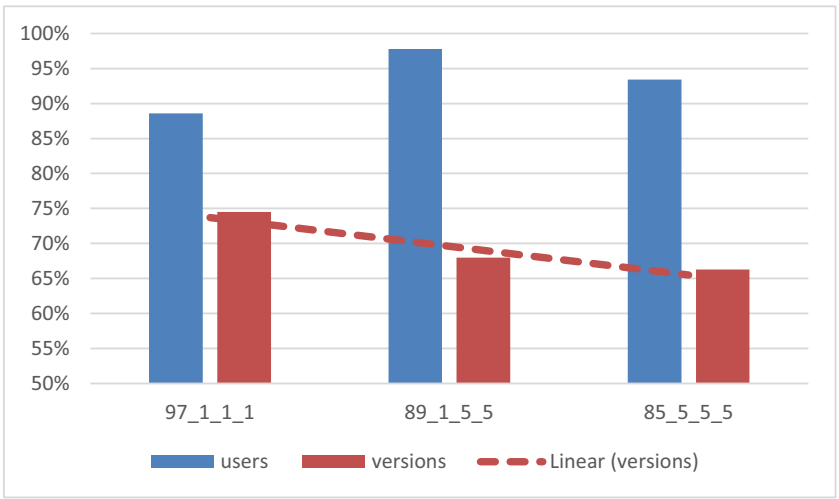

Fig. 7 Identification rates with three different distributions of activities

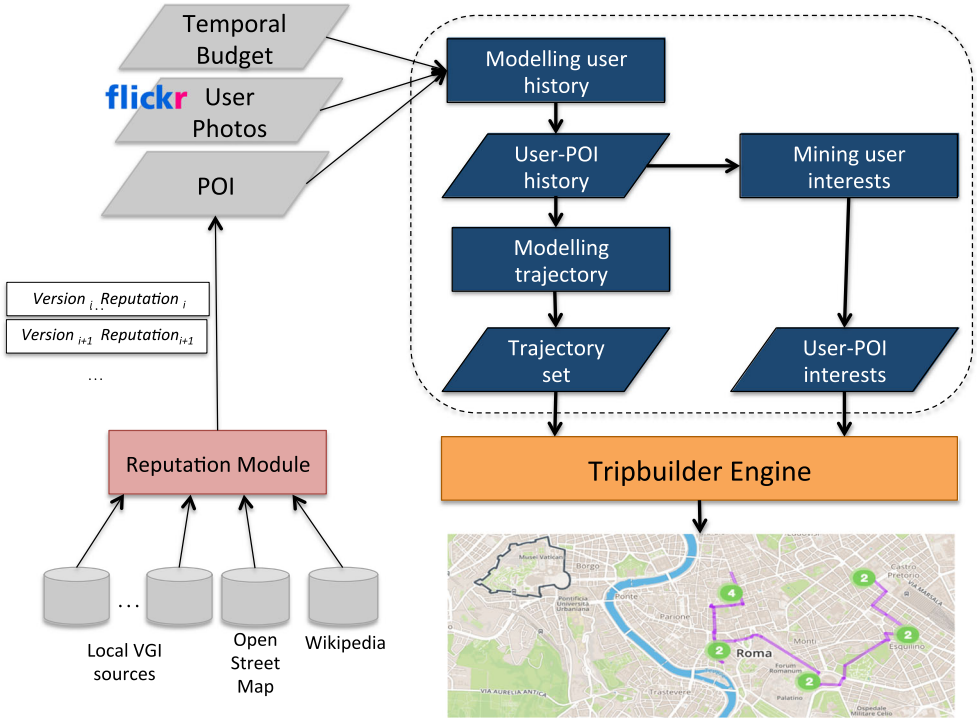

Fig. 8 TRIPBUILDER architecture including the interaction with the reputation module

6 Applicative scenario: user-generated data for tourism planning

Tourists approaching their destination for the first time deal with the problem of planning a sightseeing itinerary that covers the most subjectively interesting attractions while fitting the time available for their visit. TRIPBUILDER is an unsupervised system helping tourists to build their own personalized sightseeing tour described in [5]. Given the target destination, the time available for the visit, and the tourist's profile, TRIPBUILDER recommends a time-budgeted tour that maximizes tourist's interests and takes into account both the time needed to enjoy the attractions and to move from one PoI to the next one. A distinctive feature of this system is that the knowledge base of attractions feeding the recommendation model is entirely and automatically extracted from publicly available crowdsourced data like Flickr and Wikipedia.

However, Wikipedia as a source of points of interest for tourism recommendations has some limitations. For example, some areas of the globe (like Latin America or some Asian countries) are not covered by a sufficient number of Wikipedia pages describing tourist attractions. PoI data sparsity does not allow one to create meaningful itineraries, as several interesting attractions may be missed by the tourist. To overcome this problem, we need to rely on alternative VGI local sources of data. Local sources (i.e., sources with data provided by locals) have also the clear advantage of describing specific characteristics of the location, like the opening hours, unplanned closure time, temporary events, and specific suggestions from locals. Tourism itinerary recommendation therefore needs the integration of these local-based, up-to-date, user-generated, and reliable VGI data sources, capable of describing local attractions into the recommendation engine. Being able to distinguish reliable data from untrusted data is therefore crucial for supporting effective usage of recommendation tools like TRIPBUILDER.

In this case, it is crucial to assess the validity of these citizen-generated data dynamically.[6] This problem can be tackled in a tightly coupled architecture where TRIPBUILDER (see Fig. 8) interacts with a reputation module to assess the reputation level of the information associated with the points of interests to be recommended to the user. The attractions are represented as points of interests and the tourist traveling behavior is captured by time-stamped, geotagged Flickr photos. These traces, defined by the user behavior, are combined with the attraction information into user-PoI traces, that are mined to extract the user interests (e.g., user-preferred categories of PoI, like monuments, churches, museums, natural parks, etc.). Trajectories are then extracted by splitting the user-PoI traces into single itineraries (e.g., splitting the traces by day). Given the set of users trajectories, a time budget (e.g., 1, 2 days) and the user preferences, the TRIPBUILDER engine constructs the recommended itinerary satisfying the user preferences and the temporal budget. This is done by instantiating and

[6] Rise of the citizen scientist, http://www.nature.com/news/rise-of-the-citizen-scientist-1.18192

solving a coverage problem and a travelling salesman problem (TSP), as detailed in [5].

Reputation scores of versions describing attractions in the area of interest are transmitted to the PoI module (see Fig 8). In this module, one version among the set of the received versions is chosen (most recent, or higher reputation score). When the version reputation score of a PoI goes below a threshold, the recommendation engine excludes the PoI version from the suggested itinerary.

In Fig. 9, we see an example of a recommended path in the city of Pisa. On the top part of the figure, we see the blue PoIs as the Wikipedia entities as they represent the well-known "Pisa Leaning Tower" and "Piazza dei Miracoli," In this case, TripBuilder does not use any local VGI information but only authoritative data sources. On the bottom of the figure, we highlight a different path that takes into account additional attractions captured by local VGI information, for example, "Orto Botanico" (i.e., botanic garden). In this case, we see in dark red the attractions whose version has a reputation score higher than a fixed threshold and are included in the new path. In green color, we see local VGI PoI whose version's reputation is below the threshold, thereby not included in the recommended path. The recommendation engine reacts to the version reputation score dynamically by either adding or removing PoI based on their reputation values. This approach combines the maximum benefit in the tourist recommendation by adding local information to the popular paths while guaranteeing the reliability of this crowdsourced information.

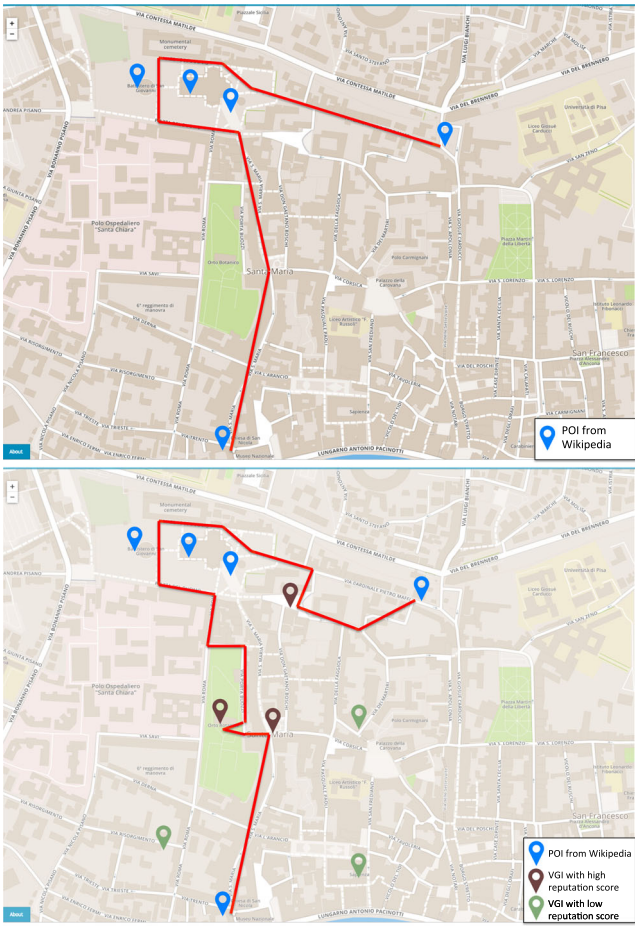

Fig. 9 Example of TripBuilder recommended paths varying based on the reputation score of the PoIs. Shown on the top of the figure is the path recommended by exploiting only Wikipedia-based PoI information. Displayed on the bottom of the figure, we see the recommended itinerary including also local VGI content. These PoIs are considered in the itinerary when version reputation is above a fixed threshold (dark red) and excluded when the score is below the threshold (green)

7 Conclusions and perspectives

Volutereed geographical information as the process of collecting, organizing, and publishing georeferenced information may provide very detailed and valuable content that can be exploited in real applications and services. This process follows modalities and principles that are typical of the crowdsourced information, but it is specialized to the geospatial domain. Estimating the quality of data provided by volunteers in this context is currently a research topic.

To this end, we presented in this paper a novel comprehensive model for reputation evaluation of VGI content presented in the context of a multi-layer architecture for VGI. The model has been evaluated preliminarily based on a variety of complex scenarios which have been simulated and run. This model relies on users' activities and it takes into account the aging of descriptions and the changes of status in real-world objects reported by VGI. A variety of requirements and metrics, as well as an algorithm to compute composite reputation scores for descriptions authored by unknown contributors, have been proposed.

The approach discussed is applicable to data sources implementing a versioning mechanism in order to keep the history of descriptions referred to the same real object/PoI. The actual computation of reputation scores is based on the analysis and count of both explicit (i.e., feedbacks) and implicit evaluations expressed by users. Our extensive experimentations showed satisfactory accuracies of the proposed model (e.g., the assignment of correct scores to versions, in the most complex scenario, was about 74%).

A possible extension of the current proposal is therefore to complement and integrate it with outcomes of techniques for evaluating VGI data quality using such kind of domain-specific knowledge, as in [7], and techniques using information on provenance of versions, as in [14].

In general, the framework for reputation evaluation needs a deeper evaluation by studying its positive features and its limits in further realistic scenarios.

Acknowledgements The Authors are thanking the colleague Prof. Gianfranco Lamperti for his very valuable help in supporting the work described in this paper, and the former student Eng. Marco Gusmini for developing the first prototypes of the proposed solution.

References

1. Aroyo L, Welty C (2013) Crowd truth: harnessing disagreement in crowdsourcing a relation extraction gold standard. WebSci2013 ACM 2013
2. Bishr M, Kuhn W (2013) Trust and reputation models for quality assessment of human sensor observations. In: Tenbrink T, Stell J, Galton A, Wood Z (eds) Spatial information theory: 11th international conference, COSIT 2013, Scarborough, UK, Proceedings. Springer International Publishing, Cham, pp 53–73. https://doi.org/10.1007/978-3-319-01790-7_4
3. Bordogna G, Carrara P, Criscuolo L, Pepe M, Rampini A (2014) A linguistic decision making approach to assess the quality of volunteer geographic information for citizen science. Inf Sci 258:312–327. https://doi.org/10.1016/j.ins.2013.07.013
4. Boulekrouche B, Jabeur N, Alimazighi Z (2016) Toward integrating grid and cloud-based concepts for an enhanced deployment of spatial data warehouses in cyber-physical system applications. J Ambient Intell Human Comput 7(4):475–487. https://doi.org/10.1007/s12652-016-0376-1
5. Brilhante IR, Macedo JA, Nardini FM, Perego R, Renso C (2015) On planning sightseeing tours with Tripbuilder. Inf Process Manag 51(2):1–15. https://doi.org/10.1016/j.ipm.2014.10.003
6. Brovelli MA, Minghini M, Zamboni G (2016) Public participation in GIS via mobile applications. ISPRS J Photogramm Remote Sens 114:306–315. https://doi.org/10.1016/j.isprsjprs.2015.04.002
7. D'Antonio F, Fogliaroni P, Kauppinen T (2014) VGI edit history reveals data trustworthiness and user reputation. In: 17th AGILE international conference on geographic information Science (Short Paper)
8. Dorn H, Törnros T, Zipf A (2015) Quality evaluation of VGI using authoritative data—a comparison with land use data in southern Germany. ISPRS Int J Geo-Inf 4(3):1657–1671
9. Goodchild MF (2007) Citizens as sensors: the world of volunteered geography. GeoJournal 69(4):211–221
10. Goodchild MF, Li L (2012) Assuring the quality of volunteered geographic information. Spat Stat 1:110–120. https://doi.org/10.1016/j.spasta.2012.03.002
11. Guo B, Wang Z, Yu Z, Wang Y, Yen NY, Huang R, Zhou X (2015) Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm. ACM Comput Surv 48(1):7:1–7:31. https://doi.org/10.1145/2794400
12. Jokar AJ, Mooney P, Zipf A, Schauss A (2015) Quality assessment of the contributed land use information from openstreetmap versus authoritative datasets. In: OpenStreetMap in GIScience. Springer, pp 37–58
13. Jung JJ (2017) Computational collective intelligence with big data: challenges and opportunities. Fut Gener Comput Syst 66 (Supplement C):87–88. https://doi.org/10.1016/j.future.2016.08.021
14. Keßler C, Trame J, Kauppinen T (2011) Tracking editing processes in volunteered geographic information: the case of openstreetmap. Identifying objects, processes and events in spatio-temporally distributed data (IOPE), workshop at conference on spatial information theory, p 12
15. Mooney P, Corcoran P, Ciepluch B (2013) The potential for using volunteered geographic information in pervasive health computing applications. J Ambient Intell Human Comput 4(6):731–745. https://doi.org/10.1007/s12652-012-0149-4
16. Rodriguez MA, Egenhofer MJ (2003) Determining semantic similarity among entity classes from different ontologies. IEEE Trans Knowl Data Eng 15(2):442–456. https://doi.org/10.1109/TKDE.2003.1185844
17. Senaratne H, Mobasheri A, Loai Ali A, Capineri C, Haklay M (2016) A review of volunteered geographic information quality assessment methods. International Journal of Geographical Information Science https://doi.org/10.1080/13658816.2016.1189556
18. Sherchan W, Nepal S, Paris C (2013) A survey of trust in social networks. ACM Comput Surv 45(4):47:1–47:33. https://doi.org/10.1145/2501654.2501661
19. Touya G, Antoniou V, Olteanu-Raimond AM, Van Damme MD (2017) Assessing crowdsourced poi quality: combining methods based on reference data, history, and spatial relations. ISPRS Int J Geo-Inf 6(3):1–29
20. Veregin H (1999) Data quality parameters. Geogr Inf Syst 1:177–189
21. Xiao C, Freeman DM, Hwa T (2015) Detecting clusters of fake accounts in online social networks. In: Proceedings of the 8th ACM workshop on artificial intelligence and security, ACM, New York, AISec '15, pp 91–101. https://doi.org/10.1145/2808769.2808779
22. Yao J, Tan W, Nepal S, Chen S, Zhang J, Roure DD, Goble C (2015) Reputationnet: reputation-based service recommendation for e-science. IEEE Trans Serv Comput 8(3):439–452. https://doi.org/10.1109/TSC.2014.2364029
23. Ye B, Wang Y (2016) Crowdrec: Trust-aware worker recommendation in crowdsourcing environments. In: 2016 IEEE international conference on web services (ICWS), pp 1–8. https://doi.org/10.1109/ICWS.2016.10
24. Yu B, Singh MP (2002) An evidential model of distributed reputation management. In: Proceedings of the first international joint conference on autonomous agents and multiagent systems: Part 1. ACM, pp 294–301
25. Yu H, Shen Z, Leung C (2013) Bringing reputation-awareness into crowdsourcing. In: 2013 9th international conference on information, communications signal processing, pp 1–5. https://doi.org/10.1109/ICICS.2013.6782912
26. Yu H, Shen Z, Leung C, Miao C, Lesser VR (2013) A survey of multi-agent trust management systems. IEEE Access 1:35–50. https://doi.org/10.1109/ACCESS.2013.2259892
27. Zhao Y, Zhou X, Li G, Xing H (2016) A spatio-temporal VGI model considering trust-related information. ISPRS Int J Geo-Inf 5(2):10